

CONTROLADOR JAVA DE LA SECRETARÍA DE ESTADO DE ADMINISTRACIONES PÚBLICAS PARA EL DNIE



Manual del Usuario

Índice

1. REQUISITOS DEL PROVEEDOR DE SEGURIDAD JCE/JCA	4
2. USO DEL PROVEEDOR	5
2.1. Funcionalidad de almacén de claves y certificados (<i>KeyStore</i>)	5
2.1.1. Solicitud de PIN para acceso a los certificados de usuario del DNle	6
2.1.2. Gestión de los diálogos gráficos de solicitud de PIN	6
2.1.3. Modo rápido del proveedor	7
2.2. Funcionalidad de firma electrónica (<i>Signature</i>)	8
2.2.1. Algoritmos soportados	9
2.2.2. Confirmación de operación de firma digital	10
2.3. Notas importantes sobre el uso	10
2.3.1. Consideraciones sobre el entorno operacional	10
2.3.2. Gestión de solicitud de PIN y confirmación de operación de firma	10
2.3.3. Posibles errores de Java	11
2.3.4. Requisitos para la validación de firmas electrónicas en Java	11
3. Excepciones que pueden ser lanzadas por el proveedor	12
3.1. Excepciones no manejadas (<i>unchecked</i>)	12
3.1.1. es.gob.jmulticard.ui.passwordcallback.CancelledOperationException	12
3.1.2. es.gob.jmulticard.ui.passwordcallback.NoConsoleException	12
3.1.3. java.security.ProviderException	12
3.1.4. java.lang.SecurityException	12
3.2. Excepciones manejadas (<i>checked</i>)	12
3.2.1. es.gob.jmulticard.jse.provider.AuthenticationModeLockedException	12
3.2.2. es.gob.jmulticard.card.BurnedDnieCardException	13
3.2.3. es.gob.jmulticard.card.InvalidCardException	13
3.2.4. es.gob.jmulticard.apdu.connection.CardNotPresentException	13
3.2.5. es.gob.jmulticard.apdu.connection.NoReadersFoundException	13
4. ACCESIBILIDAD	14
5. EJEMPLO DE USO	16
5.1. Ejemplo de extracción de certificados del titular	16
5.2. Ejemplo de realización de firma electrónica con DNle	16

5.3. Ejemplo de realización de firma electrónica sin DNle	16
5.4. Ejemplo de realización y verificación de firma electrónica	17
5.5. Ejemplo de extracción de certificados en modo rápido	17
6. GLOSARIO.....	19

NOTA:

El Controlador Java de la Secretaría de Estado de Administraciones Públicas para el DNle, desde el punto de vista de JAVA, se despliega como un Proveedor de Servicios Criptográficos (CSP), por lo que será referenciado como “Proveedor” a lo largo de este documento.

1. REQUISITOS DEL PROVEEDOR DE SEGURIDAD JCE/JCA

Para funcionar adecuadamente, es necesario lo siguiente:

- Entorno de ejecución de Java compatible con JSE (*Java Standard Edition*) versión 6 o superior, recomendándose 6 update 32 o superior.
- Un lector de tarjetas inteligentes compatible con DNle con su controlador adecuadamente instalado en el sistema operativo.
 - Consulte la documentación de su lector, de su sistema operativo y de su entorno de ejecución de Java para cerciorarse de la completa compatibilidad de todas las partes.
- Un DNle (DNI Electrónico) en buen estado de funcionamiento
 - Tenga en cuenta que unos contactos sucios, un chip desprendido o una avería interna del chip pueden causar fallos en el proveedor.

2. USO DEL PROVEEDOR

A continuación se detallan las funcionalidades del proveedor.

2.1. Funcionalidad de almacén de claves y certificados (*KeyStore*)

El proveedor se utiliza como cualquier otro proveedor de almacén de claves y certificados de Java, pero atendiendo a ciertas peculiaridades:

- El nombre declarado del almacén es DNI
- El proveedor gestiona internamente la solicitud del PIN del DNle, por lo que:
 - En la carga, se ignorará cualquier contraseña o *PasswordCallback* indicada.
 - El PIN se solicitará mediante un diálogo Swing, a menos que el entorno de ejecución no disponga de entorno gráfico, en cuyo caso se realizará una petición textual por consola.
- El proveedor implementa los siguientes métodos del interfaz [KeyStore](#) (en caso de métodos sobrecargados, se implementan los métodos con todos los posibles parámetros):
 - aliases
 - containsAlias
 - getCertificate
 - getCertificateAlias
 - getCertificateChain
 - getEntry
 - getKey
 - getProvider
 - getType
 - isCertificateEntry
 - isKeyEntry
 - load
 - size

Un ejemplo típico de carga podría ser:

```
Provider p = new DnieProvider();  
Security.addProvider(p);  
KeyStore ks = KeyStore.getInstance("DNI");  
ks.load(null, null);
```

A partir de este punto, el proveedor estaría listo para la gestión de certificados y claves digitales. Por ejemplo, para obtener el certificado (con clave pública) de firma digital del DNI podríamos usar:

```
X509Certificate cert= (X509Certificate) ks.getCertificate("CertFirmaDigital");
```

Puede encontrar más información sobre la clase *KeyStore* y el uso de proveedores de Seguridad en la documentación *JavaDoc* de Java:

- <http://docs.oracle.com/javase/6/docs/api/java/security/KeyStore.html>
- <http://docs.oracle.com/javase/6/docs/api/java/security/Security.html>
- <http://docs.oracle.com/javase/6/docs/api/java/security/Provider.html>

Y en las siguientes páginas Web:

- <http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136007.html>
- <http://docs.oracle.com/javase/6/docs/technotes/guides/security/>

2.1.1. Solicitud de PIN para acceso a los certificados de usuario del DNle

Como medida de seguridad para el usuario titular del DNle, cuando sea necesario presentar el PIN a la tarjeta para el acceso a los certificados de usuario se mostrará un cuadro de diálogo al usuario (en caso de no disponer de entorno gráfico, el diálogo se mostrará a través de la consola del sistema) notificando que se necesita la introducción del PIN para el acceso a los certificados.

A continuación se muestra el cuadro de diálogo que se mostrará al usuario para la introducción del PIN:



Nótese que una vez presentado el PIN a la tarjeta y mientras el canal seguro establecido con el DNle no se pierda, el acceso a los certificados del usuario titular del DNle o la realización de operaciones de firma digital ya no requerirán la introducción del PIN.

2.1.2. Gestión de los diálogos gráficos de solicitud de PIN

Como el proveedor gestiona de forma interna los diálogos gráficos de solicitud de PIN, el establecimiento de ciertas opciones de estos debe hacerse por medios específicos:

2.1.2.1. Establecimiento del componente padre para la modalidad del diálogo.

Puede establecer el componente padre para la modalidad de los diálogos gráficos (y evitar así que estos queden ocultos tras otra ventana) mediante la siguiente llamada estática:

```
es.gob.jmulticard.ui.passwordcallback.PasswordCallbackManager.setDialogOwnerFrame(Frame frame);
```

Donde *frame* es el componente padre, que debe ser un tipo o subtipo de *java.awt.Frame*.

El mismo componente padre establecido de esta forma será utilizado igualmente para los diálogos informativos y de confirmación (confirmación de operación de firma, etc.).

2.1.2.2. Establecimiento del idioma de los mensajes de solicitud de PIN

Puede establecer el idioma de los mensajes de solicitud de PIN (tanto en diálogo gráfico como por consola) estableciendo la siguiente propiedad del sistema con un valor textual definido en la norma ISO 639: *es.gob.jmulticard.locale*.

Un ejemplo de establecimiento de esta propiedad podría ser:

```
System.setProperty("es.gob.jmulticard.locale", "fr");
```

Es muy importante que se establezca esta propiedad antes de cargar el proveedor. Puede encontrar una lista completa de los códigos de idioma ISO 639 en: http://www.loc.gov/standards/iso639-2/php/code_list.php.

2.1.3. **Modo rápido del proveedor**

El proveedor dispone de un modo especial de funcionamiento que permite, en ciertas ocasiones, mejorar la rapidez de acceso al almacén y disminuir la cantidad de veces que se solicita el PIN del DNle al titular.

Este método rápido funciona accediendo al fichero CDF del DNle donde se encuentra información sobre los certificados disponibles en la tarjeta en lugar de realizar la lectura estándar de certificados, operación para la que es necesario introducir el PIN.

En modo rápido se construirán, por tanto, objetos Java que extienden el tipo [X509Certificate](#) pero incompletos (no contienen la codificación X.509 del certificado del titular) cuando se realiza una llamada a los métodos:

- [Certificate](#) [getCertificate](#)([String](#) alias)
- [Certificate\[\]](#) [getCertificateChain](#)([String](#) alias)

Estos certificados incompletos sólo incorporan información de las cabeceras de los certificados reales del titular e implementan sólo los siguientes métodos de la clase [X509Certificate](#):

- boolean[] [getKeyUsage](#)()
- [BigInteger](#) [getSerialNumber](#)()
- [String](#) [getSigAlgName](#)()
- [String](#) [getSigAlgOID](#)()
- [X500Principal](#) [getSubjectX500Principal](#)()
- [X500Principal](#) [getIssuerX500Principal](#)()
- int [getVersion](#)()

Una vez que se realiza una petición de la clave privada de alguno de los certificados del titular, el modo rápido queda entonces desactivado y cuando se solicita alguno de los certificados del titular, el proveedor realizará la lectura de los certificados del DNle y se retornarán entonces los certificados completos encapsulados en objetos [X509Certificate](#). Se puede realizar una petición de clave privada mediante alguno de los siguientes métodos:

- [Key](#) [getKey](#)([String](#) alias, char[] password)

- [KeyStore.Entry](#) **getEntry**(String alias, [KeyStore.ProtectionParameter](#) protParam)

El modo rápido es especialmente útil para aquellas aplicaciones que deseen mostrar información sobre los certificados del DNle sin necesidad de que el titular introduzca el PIN. Una de las acciones más comunes es mostrar al usuario, en una aplicación de firma electrónica, de qué certificados dispone para realizar una operación de este tipo, pudiendo ofrecer al usuario información más información sobre los certificados, como puede ser el Subject, número de serie, etc.

Este modo rápido viene por defecto desactivado, siendo necesario para activarlo establecer la propiedad *es.gob.jmulticard.fastmode* al valor true.

Puede establecer esta propiedad programáticamente desde Java:

```
System.setProperty("es.gob.jmulticard.fastmode", "true");
```

O desde la línea de comandos al arrancar la máquina virtual de Java:

```
java -Des.gob.jmulticard.fastmode=true
```

Puede encontrar más información sobre las propiedades del sistema en Java en la siguiente dirección Web:

- <http://docs.oracle.com/javase/tutorial/essential/environment/sysprop.html>

2.2. Funcionalidad de firma electrónica (*Signature*)

El proveedor se utiliza mediante la clase *Signature*, como en cualquier otro proveedor de firmas electrónicas de Java, pero atendiendo a ciertas peculiaridades:

- El proveedor es capaz únicamente de operar con instancias de clave privada que hereden de *es.gob.jmulticard.card.dnie.DniePrivateKey*, que únicamente son proporcionadas por él.
- No es posible obtener la codificación de las claves privadas, ya que estas no se extraen del DNle en ningún momento.
- Las operaciones con clave pública (principalmente verificaciones de firmas electrónicas) se delegan en los otros proveedores disponibles en el entorno de ejecución de Java.
 - Puede consultar información sobre los proveedores de seguridad incluidos en los entornos de ejecución de Java de Oracle en: <http://docs.oracle.com/javase/6/docs/technotes/guides/security/SunProviders.html>
- El proveedor implementa los siguientes métodos del interfaz *Signature* (en caso de métodos sobrecargados, se implementan los métodos con todos los posibles parámetros):
 - `getAlgorithm`
 - `getProvider`
 - `initSign`
 - `initVerify`
 - `sign`
 - `update`

- verify

El método `initSign` es el encargado de iniciar el proceso de creación de firma electrónica. En función del manejador de clave privada que se proporcione como parámetro, dicho método inicializará el proceso de firma electrónica haciendo uso del DNle (tras el establecimiento del canal seguro), o bien sin hacer uso del mismo. Concretamente, atendiendo a la sintaxis:

```
public final void initSign(PrivateKey privateKey)

    throws InvalidKeyException

Initialize this object for signing. If this method is called again with a
different argument, it negates the effect of this call.

Parameters:

    privateKey - the private key of the identity whose signature is going
to be generated.

Throws:

    InvalidKeyException - if the key is invalid.
```

Si el parámetro `privateKey` es un manejador de una clave privada almacenada en el DNle, se invocará la funcionalidad de firma del DNle a través del canal seguro establecido con el mismo. (**Ver Ejemplo 5.2**). Si por el contrario el parámetro `privateKey` es una clave privada distinta a las albergadas en el DNle (por ejemplo, la clave privada de un par de claves previamente generado), la firma se realizará sin hacer uso del DNle, y por tanto no se construirá ningún canal seguro con él. (**Ver ejemplo 5.3**).

Cabe destacar que el Proveedor se ejercita únicamente si el parámetro `privateKey` de la función `initSign` es un manejador de una clave privada del DNle, pues en caso contrario será el entorno quien ejecute la acción de firma electrónica. Por razones de claridad, se incluyen ejemplos de invocación del método `initSign` tanto con manejadores de claves privadas del DNle (ejemplo 5.2) como con claves privadas ajenas al mismo (ejemplo 5.3), aunque, como se ha indicado, únicamente el ejemplo 5.2 invoca funcionalidad del TOE.

Dado que el servicio de firma electrónica únicamente realiza operaciones de clave privada con las clases de clave privada proporcionadas por el mismo proveedor (con los procedimientos descritos en el punto 2.1 de este mismo documento), es necesario el uso combinado de ambas funcionalidades. Consulte el punto 3 del documento para un ejemplo completo de uso.

Puede encontrar más información sobre la clase *Signature* de Java en:

- <http://docs.oracle.com/javase/6/docs/api/java/security/Signature.html>

2.2.1. Algoritmos soportados

El proveedor soporta los siguientes algoritmos de firma electrónica:

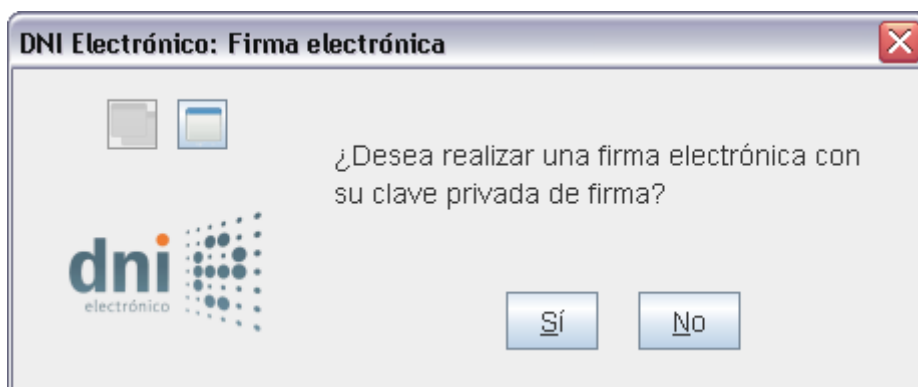
- SHA1withRSA
- SHA-256withRSA
- SHA-384withRSA

- SHA512withRSA

El proveedor no acepta claves privadas de otros proveedores para realizar las operaciones de firma electrónica.

2.2.2. Confirmación de operación de firma digital

Como medida de seguridad para el usuario titular del DNle, toda vez que se solicite al proveedor de seguridad la realización de una operación criptográfica de firma, se mostrará un cuadro de diálogo al usuario (en caso de no disponer de entorno gráfico, el diálogo se mostrará a través de la consola del sistema) notificando la inminente realización de la acción, suministrando además información sobre el certificado concreto del DNle que utilizará para la realización de la operación de firma electrónica, ya sea el certificado de autenticación o el de firma.



Como se ve en la figura anterior, el usuario titular del DNle tiene la posibilidad de continuar con la operación de firma o cancelarla.

2.3. Notas importantes sobre el uso

2.3.1. Consideraciones sobre el entorno operacional

Existen una serie de aspectos relevantes a tener en cuenta sobre el entorno en el que se va a ejecutar el proveedor de seguridad. Dichas cuestiones se pueden consultar en el “*Manual de Instalación*”, dentro del apartado 1 “*Requisitos del proveedor de seguridad JCE/JCA*”.

2.3.2. Gestión de solicitud de PIN y confirmación de operación de firma

Como medida de seguridad, el controlador se responsabiliza de realizar la solicitud de PIN al usuario así como de la solicitud de confirmación para la realización de una operación de firma electrónica.

Para ello, la lógica que sigue el controlador a la hora de mostrar los diálogos, tanto de solicitud de PIN como de confirmación de operación de firma, es la siguiente:

1. Petición mediante diálogos gráficos: Se trata del comportamiento por defecto. En caso de que el controlador se ejecute bajo un entorno que disponga de modo gráfico, se mostrarán al usuario los diálogos gráficos ilustrados en los puntos 2.1.1 y 2.2.2 respectivamente.

2. Petición mediante diálogos por consola de texto: En caso de que no se disponga de modo gráfico, se mostrarán por consola de texto mensajes que soliciten al usuario la introducción del PIN o de confirmación para la realización de una operación de firma.

En un entorno con modo gráfico, sería posible forzar esta situación estableciendo, por ejemplo, la siguiente propiedad de Java a *true*:

```
System.setProperty("java.awt.headless", "true");
```

En el hipotético caso de que el controlador no pudiese mostrar los diálogos gráficos ni mediante el uso de la consola de texto, éste lanzaría una excepción de tipo *java.lang.IllegalStateException* indicando que *"No hay consola para solicitar el PIN"*.

2.3.3. Posibles errores de Java

Ciertas versiones del JRE (especialmente las anteriores a la 6 update 27) tienen un error en el API de acceso a los lectores de tarjetas (JRS-268, SmartCardIO) que causa importantes problemas de funcionamiento en el proveedor cuando se usa en un entorno con varios hilos de ejecución.

Como precaución, intente limitar en la medida de lo posible las aplicaciones que hagan uso del proveedor a un único hilo de ejecución.

Se puede encontrar más información en los siguientes enlaces:

- http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6963006
- http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=2209222
- http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=2209883

2.3.4. Requisitos para la validación de firmas electrónicas en Java

Para la verificación de firmas electrónicas, JSE requiere que si se hacen indicando un certificado y este tiene marcado como extensión crítica el *KeyUsage*, este sea compatible con firmas digitales. Dado que este requisito no se cumple con el certificado de firma del DNle, para realizar una validación de una firma realizada con la clave privada de firma será necesario indicar directamente la clave pública, y nunca directamente el certificado completo.

Más información en:

[http://docs.oracle.com/javase/6/docs/api/java/security/Signature.html#initVerify\(java.security.cert.Certificate\)](http://docs.oracle.com/javase/6/docs/api/java/security/Signature.html#initVerify(java.security.cert.Certificate))

3. Excepciones que pueden ser lanzadas por el proveedor

El proveedor puede lanzar una serie de excepciones, propias y/o de carácter genérico, para indicar situaciones de excepción relacionadas directa o indirectamente con el uso del DNle mediante el proveedor.

3.1. Excepciones no manejadas (*unchecked*)

3.1.1. `es.gob.jmulticard.ui.passwordcallback.CancelledOperationException`

Esta excepción se lanza tanto desde el *KeyStore* como desde el *Signature* cuando la operación en curso se ha cancelado por una causa atribuible al usuario. En el proveedor, su aparición es indicativa de que el usuario ha rechazado un diálogo, de introducción de PIN o de confirmación de operación de firma.

3.1.2. `es.gob.jmulticard.ui.passwordcallback.NoConsoleException`

Esta excepción se lanza cuando el controlador no es capaz de obtener la consola del sistema para mostrar al usuario la petición del PIN, ya se trate de una ejecución en un entorno de modo texto o el modo gráfico no esté disponible.

3.1.3. `java.security.ProviderException`

Esta excepción se lanza cuando se produce un error irrecuperable interno al proveedor. Un ejemplo de este tipo de error es la detección de APDUs modificadas una vez establecido canal seguro con el DNle. En todo caso, se añade información para facilitar la identificación de la causa del error producido.

3.1.4. `java.lang.SecurityException`

El proveedor está empaquetado en un fichero JAR firmado. En caso de que se realice alguna modificación sobre alguno de los ficheros empaquetados en el JAR, la JVM detectará en tiempo de ejecución que esos ficheros han sido modificados y lanzará una excepción de este tipo indicando que la firma de esos ficheros no coincide con la esperada.

3.2. Excepciones manejadas (*checked*)

3.2.1. `es.gob.jmulticard.jse.provider.AuthenticationModeLockedException`

Esta excepción hereda de *SignatureException* e indica que el DNle está bloqueado por exceder el límite máximo de intentos al introducir el PIN. Puede ser lanzada al solicitar la firma desde el *Signature*, y siempre tras una introducción de PIN por parte del usuario.

Esta excepción puede ser la causa de una *ProviderException* al recuperar un certificado o una entrada desde el *KeyStore*.

3.2.2. es.gob.jmulticard.card.BurnedDnieCardException

Esta excepción hereda de *IOException* e indica que se ha encontrado un DNle en el lector de tarjetas, pero que este tiene su memoria volátil borrada y por lo tanto no se pueden realizar operaciones criptográficas con él. Se lanza desde el *KeyStore* en el momento de su carga.

3.2.3. es.gob.jmulticard.card.InvalidCardException

Esta excepción hereda de *IOException* e indica que se ha encontrado una tarjeta inteligente en el lector, pero que esta no es un DNle y por lo tanto el proveedor no puede operar con ella. Se lanza desde el *KeyStore* en el momento de su carga.

3.2.4. es.gob.jmulticard.apdu.connection.CardNotPresentException

Esta excepción hereda de *IOException* e indica que se no ha encontrado ninguna tarjeta inteligente en el lector, y por lo tanto el proveedor no puede operar. Se lanza desde el *KeyStore* en el momento de su carga. En algunos sistemas no es posible detectar esta situación (por ejemplo, Mac OS X), por lo que se lanzará una excepción genérica de error en la comunicación con la tarjeta

3.2.5. es.gob.jmulticard.apdu.connection.NoReadersFoundException

Esta excepción hereda de *IOException* e indica que no se han detectado lectores de tarjetas inteligentes en el sistema, por lo que el proveedor no puede operar. Se lanza desde el *KeyStore* en el momento de su carga.

4. ACCESIBILIDAD

El Controlador Java de la Secretaría de Estado para las Administraciones Públicas para el DNle se ha diseñado e implementado utilizando la norma de accesibilidad UNE 139802:2009, de modo que su comportamiento cumple los aspectos determinados por la norma anterior. Las aplicaciones que incluyan el Controlador ofrecerán, por lo tanto, estas características de accesibilidad que facilitan la interacción con los elementos del controlador, de modo que deberán informar al usuario de ello en su documentación. Para informar de un modo adecuado de dichas características se deberá incluir lo siguiente:

- *“El Controlador Java de la Secretaría de Estado para las Administraciones Públicas para el DNle está preparado para tomar automáticamente la configuración estética definida para las ventanas del sistema operativo. Por tanto, mantendrá el mismo aspecto visual establecido por el usuario para su equipo.”*

- Uso de [Java Access Bridge](#). Si se desea hacer uso de algunas herramientas como lectores o ampliadores de pantalla, será necesario que se instale previamente. Texto a incluir:

“Recuerde que el <nombre aplicación> es una aplicación Java, por lo que si desea hacer uso de algunas herramientas como algunos lectores o ampliadores de pantalla, será necesario que instale previamente [Java Access Bridge](#). Si lo desea puede consultar las [instrucciones en español de la instalación](#) de Java Access Bridge 2.0.2. Únicamente para sistemas con Windows 32 bits se podría utilizar el instalable de una versión anterior de [Java Access Bridge \(versión 2.0.1\)](#).”

- Redimensión de la ventana de solicitud de PIN y la existencia de botones para cambiar el tamaño de la ventana. El usuario puede modificar el tamaño de la ventana para adaptarla a sus necesidades ya que los elementos y textos contenidos se redimensionarán convenientemente.

“La ventana de petición de PIN al usuario puede ser redimensionada por parte de los usuarios. Los elementos contenidos en ella se redimensionarán en relación al nuevo tamaño, adaptándose a las necesidades del usuario.”

- Atajos de teclado:

- *Alt + A: Botón “Aceptar”*
- *Alt + C: Botón “Cancelar”*
- *Alt + P: Campo para introducir contraseña*
- *Alt + R: Restaurar tamaño de ventana*
- *Alt + M: Maximizar tamaño de ventana*
- *Alt + S: Botón “Sí”*
- *Alt + N: Botón “No”*

Declaración de accesibilidad:

“En la construcción del Controlador de la Secretaría de Estado para las Administraciones Públicas Java para el DNIe se han adoptado una serie de medidas con el objetivo de hacerlo accesible conforme a la norma UNE 139803:2009. Esto conlleva una serie de ventajas como:

- *Facilitar el acceso de los usuarios independientemente de su condición física o de su entorno.*
- *Permitir el acceso con diferentes agentes de usuario.*

Entre otras se han adoptado las siguientes medidas:

- *Los elementos de interfaz disponen de un texto alternativo equivalente.*
- *Los componentes del controlador se adaptan a los diferentes tamaños de ventana*
- *Se utilizan estándares de diseño e interacción.*
- *Las opciones principales incluyen acceso mediante atajos de teclado.”*

5. EJEMPLO DE USO

5.1. Ejemplo de extracción de certificados del titular

```
// Se instancia el proveedor y se anade
final Provider p = new DnieProvider();
Security.addProvider(p);

// Se obtiene el almacén y se carga
final KeyStore ks = KeyStore.getInstance("DNI"); //$NON-NLS-1$
ks.load(null, null);
Certificate authCert = ks.getCertificate("CertAutenticacion");
Certificate authCert = ks.getCertificate("CertFirmaDigital");
```

5.2. Ejemplo de realización de firma electrónica con DNle

```
// Se instancia el proveedor y se anade
final Provider p = new DnieProvider();
Security.addProvider(p);

// Se obtiene el almacén y se carga
final KeyStore ks = KeyStore.getInstance("DNI"); //$NON-NLS-1$
ks.load(null, null);

final String alias = "CertFirmaDigital"; //$NON-NLS-1$

// Se obtiene el motor de firma y se inicializa
final Signature signature = Signature.getInstance("SHA1withRSA"); //$NON-NLS-1$
signature.initSign((PrivateKey) ks.getKey(alias, null));

// Vamos a firmar el texto 'hola'
signature.update("hola".getBytes()); //$NON-NLS-1$

// Completamos el proceso y obtenemos la firma PKCS#1
final byte[] signatureBytes = signature.sign();
```

5.3. Ejemplo de realización de firma electrónica sin DNle

```
// Generamos un par de claves
KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");

kpg.initialize(1024);

KeyPair keyPair = kpg.genKeyPair();

Key publicKey = keyPair.getPublic();
```

```
Key privateKey = keyPair.getPrivate();

// Algoritmo de firma

final Signature signature = Signature.getInstance("SHA1withRSA");

// Firma
signature.initSign(keyPair.getPrivate());

// Vamos a firmar el texto 'hola'
signature.update("hola".getBytes());

final byte[] signatureBytes = signature.sign();
```

5.4. Ejemplo de realización y verificación de firma electrónica

```
// Se instancia el proveedor y se anade
final Provider p = new DnieProvider();
Security.addProvider(p);

// Se obtiene el almacen y se carga
final KeyStore ks = KeyStore.getInstance("DNI"); //$NON-NLS-1$
ks.load(null, null);

final String alias = "CertFirmaDigital"; //$NON-NLS-1$

// Se obtiene el motor de firma y se inicializa
final Signature signature = Signature.getInstance("SHA1withRSA"); //$NON-NLS-1$
signature.initSign((PrivateKey) ks.getKey(alias, null));

// Vamos a firmar el texto 'hola'
signature.update("hola".getBytes()); //$NON-NLS-1$

// Completamos el proceso y obtenemos la firma PKCS#1
final byte[] signatureBytes = signature.sign();

// Comprobacion de la validez de la firma obtenida
signature.initVerify(ks.getCertificate(alias).getPublicKey());
signature.update("hola".getBytes()); //$NON-NLS-1$
signature.verify(signatureBytes);
```

5.5. Ejemplo de extracción de certificados en modo rápido

```
// Se activa el modo rápido
final Provider p = new DnieProvider();
System.setProperty("es.gob.jmulticard.fastmode", "true");

// Se instancia el proveedor y se anade
final Provider p = new DnieProvider();
Security.addProvider(p);

// Se obtiene el almacen y se carga
```

```
final KeyStore ks = KeyStore.getInstance("DNI"); //$NON-NLS-1$
ks.load(null, null);

// El siguiente certificado es impostado
Certificate authCertImpostado = ks.getCertificate("CertAutenticacion");

// El siguiente certificado es completo/real
KeyStore.Entry entry = ks.getEntry("CertAutenticacion", null);
KeyStore.PrivateKeyEntry pkEntry = (KeyStore.PrivateKeyEntry) entry;
Certificate authCertCompleto = pkEntry.getCertificate();
```

6. GLOSARIO

- JDK
 - *Java Development Kit*, conjunto de desarrollo de Java. Es un conjunto de herramientas, clases, bibliotecas y documentación para el desarrollo de aplicaciones Java.
- JSE
 - *Java Standard Edition*. Es el subconjunto de la plataforma Java orientado a ordenadores personales y estaciones de trabajo.
- JRE
 - *Java Runtime Environment*, entorno de ejecución de Java. Consiste en una máquina virtual y las clases, bibliotecas y herramientas necesarias para la ejecución de programas Java.
- DNle
 - Documento Nacional de Identidad electrónico
- API
 - *Application Programming Interface*, interfaz de programación de aplicaciones
- PIN
 - *Personal Identification Number*, número de identificación personal
- EAL
 - *Evaluation Assurance Level*, nivel de evaluación de la confianza. Es un grado numérico referente a las normativas internacionales *Common Criteria* de seguridad.
- JAR
 - *Java Archive*, archivo Java. Es una agrupación de clases y recursos Java en un único fichero.